ITEA Journal 2008; 29: 402–410

A Sensor Network Architecture: Information, Control, and Behavior Definitions for Large-Scale or Systems-of-Systems Testing

Howard E. Michel, Ph.D. and Hemant Joshi
Electrical & Computer Engineering Department,
University of Massachusetts Dartmouth, North Dartmouth, Massachusetts

This article envisions a plug-and-play architecture for test and evaluation that will allow engineers to rapidly and robustly define and configure test environments and scenarios. The architecture described here is based on a layered functional decomposition of the three aspects of test: information flow, control flow, and behavior. These individual layered decompositions are presented first, then as an integrated technical reference model. Having an integrated technical reference model is crucial to developing affordable and robust systems that are self-aware, self-healing, and adaptable within a resource-constrained environment. Having these capabilities will become increasingly important as test scenarios become increasingly complex, such as distributed system-of-systems testing or as information volume becomes increasingly more demanding, yet unpredictable, as in the case of continuous test during operational missions. This work is presented as a natural extension of the military's Network Centric Warfare model.

Key words: Adaptable systems; autonomous sensor networks; control distribution; information processing; integrated technical reference models; intelligence; plug-and-play sensors; sensor network architecture; information processing.

Consider a test scenario with 500,000 individual sensors, where commanders on the ground, not test directors, control test execution and where the test and evaluation (T&E) tasking is to report on the effectiveness of each weapon system, including the soldier in the loop. Further consider that you have only months to plan this test, and weeks to report the results. Such might be the situation when testing the Army's future combat system in a force-on-force exercise. And simultaneously, with this tasking, you have the requirement to identify and report on any system, subsystem, or component that has produced an anomalous response, either as a result of previously unobserved combinations of environmental conditions or random situations that had not been observed in previous testing.

Or consider the task of testing swarms of autonomous air vehicles. These vehicles will exhibit adaptive, collaborative behaviors that respond to changes in the environment and their individual and cooperative capabilities. How do you define a test scenario for a continuously adapting and changing system?

Clearly, the state-of-the-art in T&E for military systems today could not handle this tasking. What is required is a catalog of nonintrusive instrumentation—sensors, processors, storage devices, and software—to continuously monitor key parameters, accompanied by user-friendly interfaces that can be used to rapidly configure these components into a system for each test. Furthermore, if this system is to be cost-effective, it should be designed around modular components that are built with open interface standards for hardware, software, and data and metadata formats. If the system is to be robust, it needs to be self-aware, self-healing, and adaptable within a resource-constrained environment.

This article proposes a methodology for describing the functionality of such a system in a manner enabling industry and the military to develop a plug-and-play catalog of sensors, processors, and software modules, allowing test engineers to easily and cheaply configure test systems to meet the requirements of future T&E.

The discussion begins with describing a model represented as a three-sided layered pyramid. The first face of this abstraction describes data and information

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2008		**00-00-2008 to 00-00-2008**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Sensor Network Architecture: Information, Control, and Behavior Definitions for Large-Scale or Systems-of-Systems Testing	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Massachusetts Dartmouth,Electrical & Computer Engineering Department,North Dartmouth,MA, 02747-230**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**9**	

flow; the second face describes the control flow; the third face represents behavior abstractions. These models are then combined into an integrated technical reference model (I-TRM) and a simple architecture based on the I-TRM is proposed. These developments are illustrated with a simple T&E example.

Models and architectures

A model is used for representing a set of components of a process, system, or subject area, or for developing, understanding, analyzing, improving, and/or replacing a process (ICH 2008). It is also used to define the meaning of common concepts, illustrate the relationships between those concepts, and assist in understanding how current and future systems fit into that model (Visnevski and Bezdecny 2008). A technical reference model (TRM) is used to formulate definitions and provide a formal structure for describing implicit and explicit concepts and operations.

One popular TRM, the International Organization for Standardization's (ISO) Open System Interconnection (OSI) TRM focused on internetwork communication by putting forward a seven-layered abstraction of the functions required in computer communications (Day and Zimmermann 1983, ISO 1983, Tannenbaum 1996). This TRM was very successful in establishing a framework for describing and developing operating concepts, but it was less successful in establishing implementable standards and products. The Internet today is based on the transmission control protocol/internet protocol (TCP/IP) architecture and associated standards (Tannenbaum 1996).

A TRM is different from an architecture. An architecture is used to describe the arrangement of system, function, and design components and interfaces that comprise a solution satisfying a set of requirements (IEEE 1999). A well designed TRM can be used in developing such an architecture.

An example of this model-to-architecture process can be found in the Embedded Instruments System Architecture program, now the Non-Intrusive Instrumentation (NII) program run by the Test and Resources Management Center's Test and Evaluation/Science and Technology Program NII office through the Naval Underwater Warfare Center.

Under contract to this program, General Electric (GE) developed the Open Modular Embedded Architecture (OMEA). The OMEA architecture is a working incarnation of the Information Centric TRM proposed by Michel and Fortier (2005) and Fortier and Michel (2005). The OMEA test suite was fully implemented on Agilent, Yokogawa, and Video instruments. OMEA also implemented synthetic instrumentation and a user interface, and was demonstrated in 2007 (Visnevski 2008).

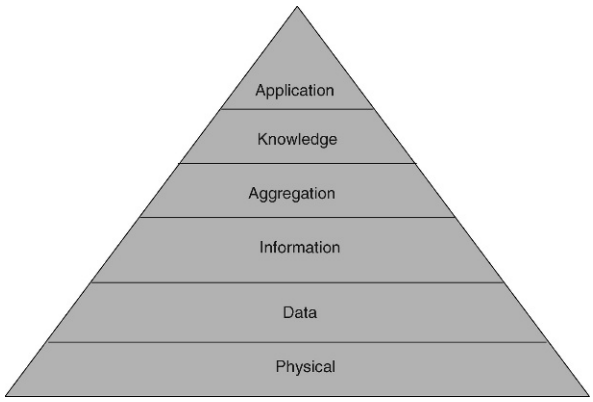

Figure 1. The information-centric face

Proposed integrated technical reference model, the I-TRM

The proposed I-TRM is composed of three faces of a pyramid that describe data, information, and knowledge flow; control flow; and behavior, respectively. The faces are composed of six layers that span their respective domain, but each has a common physical layer at the bottom and a common application at the peak of the pyramid. The data–information–knowledge face and the control face will be described in terms of transformations or functionality, but it is envisioned that the formal specification of the interfaces between these layers would be data and metadata definitions. The specification of the behavior layers is and would be in the form of algorithms. Additionally, the layers are not meant to define distinct hardware layers or software layers because engineering efficiency may dictate integrated products that span multiple layers.

The information-centric face (IC-face)

The IC-face provides a description of the functions associated with data collection, information aggregation, and knowledge generation. It does not specify details related to the control mechanism that manages how and where the data are collected. It is based on the Information-Centric Technical Reference Model (IC-TRM) proposed by Michel and Fortier (2005) and Fortier and Michel (2005) and refined by Joshi and Michel (2007, 2008).

This six-layered view of the IC-face is shown in *Figure 1* and described further on. Lower layers deal with an enormous amount of data that have very low information value. As we move up the layers, the data volume decreases but the information value of that data

increases. All data are transformed through these six layers, although some of these layers may be minimal in certain situations. The individual layers, described from the lowest layer up to the top layer, are as follows.

The physical layer. The physical layer gathers and manipulates raw data in unformatted, unverified, and transitory format and deals with the electric and mechanical characteristics of the system. It is composed of electromechanical sensors. An example of data at the physical layer might be the voltage output from a thermocouple. These data are clearly transitory and volatile. Metadata associated with the physical layer would be the sensor type, serial number, location, and calibration status. These metadata would generally exist in a stable form as part of the physical sensor.

The data layer. The data layer performs extraction and transformation of data into digital form and checks the authenticity of the measurements. In our given example, the voltage from the physical layer is transformed into a byte or a word using a prescribed (although possibly variable) process involving amplifiers, filters, and analog-to-digital converters. Variable parameters could include sampling rate, digitization accuracy, filter cutoff frequency, amplifier gain, etc. Metadata generated at this level could include these parameters, plus a time tag, a verification bit to indicate that the sensor is calibrated and operating properly, etc. Metadata from the physical layer and data layer would be bundled with the data to form an informative data packet.

The information layer. The information layer correlates data with scaling, location, type of measurement, etc., to produce information about the system or environment. Continuing with our simple scenario, the data and metadata from the data layer would be combined to produce information that reports, for example, that the temperature at the 12-o'clock position in the combustion chamber of the number one engine was 1,000°F at $T + 1.0$ seconds from test start, and that this measurement should be believed with a high degree of confidence. Notice, that the sensor serial number, location, calibration status, etc., are still available in the data, but hidden. At this level information is made available to the system and users of the system. Metadata created at this level would involve defining the state of these data-converting processes, including for example, the criteria used to define "a high degree of confidence."

The aggregation layer. The aggregation layer performs knowledge aggregation by goal-directed information merging from various sources, as per the requirements of the system or subsystem under test. Continuing now with our jet-engine example, there may be temperature sensors located at the 3-, 6-, and 9-o'clock positions in the engine combustion chamber. There probably are also temperature sensors located at the inlet and exhaust of the engine, as well as pressure sensors, fuel flow sensors, etc. Virtual instruments can be created at this level. For example, readings from multiple temperature sensors, with synchronized time tags, could be combined to give an instantaneous view of the temperature gradients within the combustion chamber. Additionally, a moving window of a time-sequenced series of readings could be combined to provide the dynamic response to changes in the system. Temperatures, pressures, and fuel flows could be combined to create a measure of engine efficiency. The aggregation layer produces information at a higher level of value and lower rate than the information layer below it. Metadata created at this layer would include information about the processes used to aggregate the information from the lower layers.

The knowledge layer. The knowledge layer transforms aggregated information into knowledge by processing it against intrinsic and extrinsic information, and knowledge available. For example, external information such as engine-temperature redline limits could be brought into the model at this level. If the engine temperature approached or exceeded this value, warnings could be issued, or commands could be issued to lower layers in the T&E system to increase sampling rate or accuracy of the engine temperature sensors so a more accurate post-test analysis could be conducted.

The application layer. The application layer concentrates on user–system interaction. It provides a means of accessing and using information for the user in a consistent format, from the system.

The control face (C-face)

The C-face of the I-TRM defines a layered abstraction for describing tasks in a hierarchy from high-level goal definition, through task validation, translation, distribution, and execution. The C-face is derived from the Control Technical Reference Model proposed by Dipple and Michel (2006) and refined by Joshi and Michel (2007, 2008). The C-face concentrates on hierarchical control and task distribution, and primarily builds on the initial work done in the field of control architecture by Albus at the National Institute of Standards and Technology (Albus et al. 1981, 1989; Albus and Ripley 1994; Barbera et al. 1982). Other

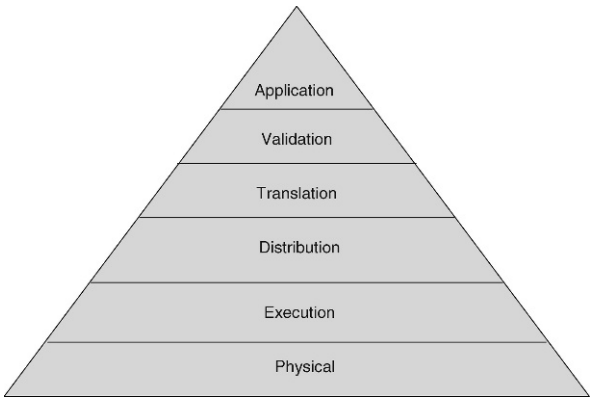

Figure 2. The control face

examples of control-centric definitions are the Mission Oriented Operating Suite (MOOS) (MOOS 2008) and the Joint Architecture for Unmanned Systems (JAUS) (JAUS 2008).

The C-face thus describes the transformation of high-level user-friendly goals into executable tasks. In following our engine performance example, it would relieve the test engineer from having to keep track of individual sensor serial numbers, exact locations, calibration history, etc., and the proper configurations and operating parameters to create synthetic instruments, etc. This information would be created automatically using the flow up of metadata and applying rules specified in defining the T&E case. It would also allow the system to retask individual sensors or groups of sensors in response to predetermined conditions, and thus possibly provide vital information in anomalous situations at the cost of not having multiple copies of slow-changing redundant data. The six layers of the C-face are shown in *Figure 2* and described in the following paragraphs, from the application layer down to the physical layer.

The application layer. The application layer focuses on user–system interaction. It provides an interface for the user to interact with the system to define mission goals. Our simple test scenario illustrating data transformation from an engine temperature sensor (described in the IC-face narrative) may be part of a larger T&E goal, producing a family of engine performance curves.

The validation layer. The validation layer provides a mechanism for authenticating the semantic correctness of the goal and for determining whether the goal is accepted or not. These processes are based on intrinsic and extrinsic information and knowledge. This layer also verifies the probability of accomplishing the goal with the resources available. In our continuing example, metadata from the lower layers would be evaluated here to assess the readiness of the T&E system to produce the family of engine performance curves requested. Are all of the required sensors in place, calibrated, and operating correctly? Is the software in place to create the required synthetic instruments? Are there other, higher priority tasks causing conflicting demands on T&E resources?

The translation layer. The translation layer decomposes valid goals into functional tasks based on knowledge about the lower layers. This layer provides a mechanism to register low-level system components and their physical capabilities. This layer would, in our example, create the need for synthetic instruments to produce synchronized temperature and pressure readings in the combustion chamber of engine one, and synchronize them with corresponding readings for fuel flow and aircraft altitude. It would pass these test requirements down to the distribution layer.

The distribution layer. The distribution layer, based on available spatial and temporal information (passed up from lower layers as metadata), organizes system tasks by decomposing the task groups into subtasks and assigning priorities to them in accordance with preestablished or dynamic goals. Continuing our example, one of the tasks received by the distribution layer might be to create an instrument to record a series of temperature readings correlated with pressure readings from engine one. Software at this level might understand, for example, that there are four individual temperature sensors in the combustion chamber, and that if they all are reading within 50°F of each other, then any one of them can be used to report the subject temperature. Furthermore, software at this level would understand that these temperature readings need to be closely synchronized with pressure and fuel flow readings, but that because of physical characteristics, synchronization at the fraction of a second level is sufficient, rather than at a microsecond level of synchronization. These commands are then passed down to the execution layer.

The execution layer. The execution layer receives directives from the distribution layer and transforms them into control signals for the physical layer. For example, the execution layer, based on its detailed understanding of the sensors involved, would issue a command to temperature sensor number XYZ to stream temperature data (assuming it has a stream mode) at 100 samples/s, using gain N starting at time T.

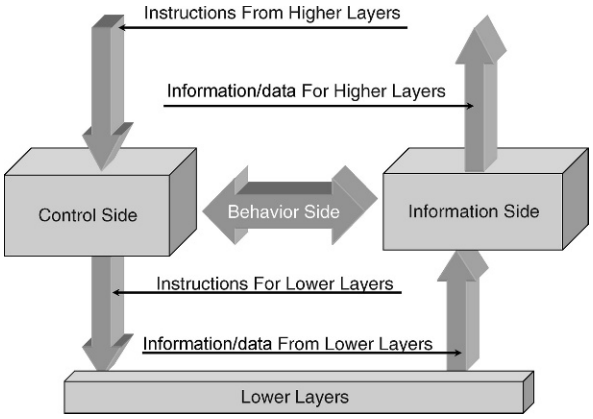

Figure 3. Control loops in the I-TRM

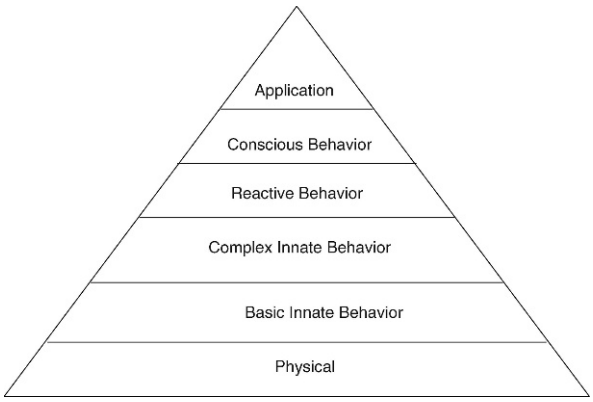

Figure 4. The behavior face

The physical layer. The physical layer constitutes sensors and mechanical units. It executes actions as directed by higher layers.

The behavioral face (B-face)

The B-face describes the intelligence (deliberative and reactive behaviors) of the system and acts as a bridge connecting the IC-face and C-face. This bridge is implemented using control loops based on a classical closed loop control system methodology (Nagrath and Gopal 1981) as adapted in Joshi and Michel (2007, 2008). As shown in *Figure 3*, commands and goals flow down, data and metadata (or information and meta-information) flow up, and the behavior at each layer interprets the execution of the commands or processes the data based on the system status as reported in the metadata or meta-information.

The B-face is a hierarchal arrangement of behaviors into layers based on the scope of control and responsibility of each function. This hierarchical distribution is based on Arkin's work, according to which behaviors can be divided into three major categories: innate behaviors, reactive behaviors, and conscious behaviors (Arkin 1998). The six layers of the B-face are illustrated in *Figure 4* and described in the following paragraphs, from the physical layer up to the application layer.

The physical layer. The physical layer constitutes sensors and mechanical units. It is the same hardware as described in the IC-face and C-face.

The basic innate behavior layer. The basic innate behavior layer implements primitive reflexive behavior and stimulus–response behaviors of the system. It combines the execution layer procedure–execution to produce the relevant data. In terms of smart sensors implied in the on-going example, one basic innate behavior would be to produce a data-word associated with a physical temperature. Another basic innate behavior would be to report metadata when queried.

The complex innate behavior layer. The complex innate behavior layer is the highest reflexive layer. It implements procedures that connect the information extracted from the data and metadata passed up with task execution distribution passed down. Complex innate behaviors are composed of one or more basic innate behaviors structured in a predefined manner to produce a higher-level user-friendly interface. An example of a complex innate behavior might be to "continuously sample and stream data at a particular rate, gain setting, and filter characteristics." Another complex innate behavior might be a self-calibration mode.

The reactive behavior layer. The reactive behavior layer provides a mechanism for dealing with information collaboration from various modules into one structured data unit (local model) (Norvales et al. 2006). It also provides procedures for translating goals into submodules in compliance with the state of the environment. This behavior requires a sophisticated understanding of the state of the various system sensors and rules to interpret the goal-oriented system tasking. This understanding and the rules would be test-specific, but easily configured from more general rule prototypes. In continuing with our aircraft engine example, creating virtual instruments that span either temporal or multisensor data streams would occur at this level. Software at this level would understand that it has various independent sources of temperature and pressure readings from the correct locations that can be correlated to create the appropriate synthetic instrument. The software would also know how to react and retask these sensors if priorities or system status changes, as might be the case if a sensor failed.

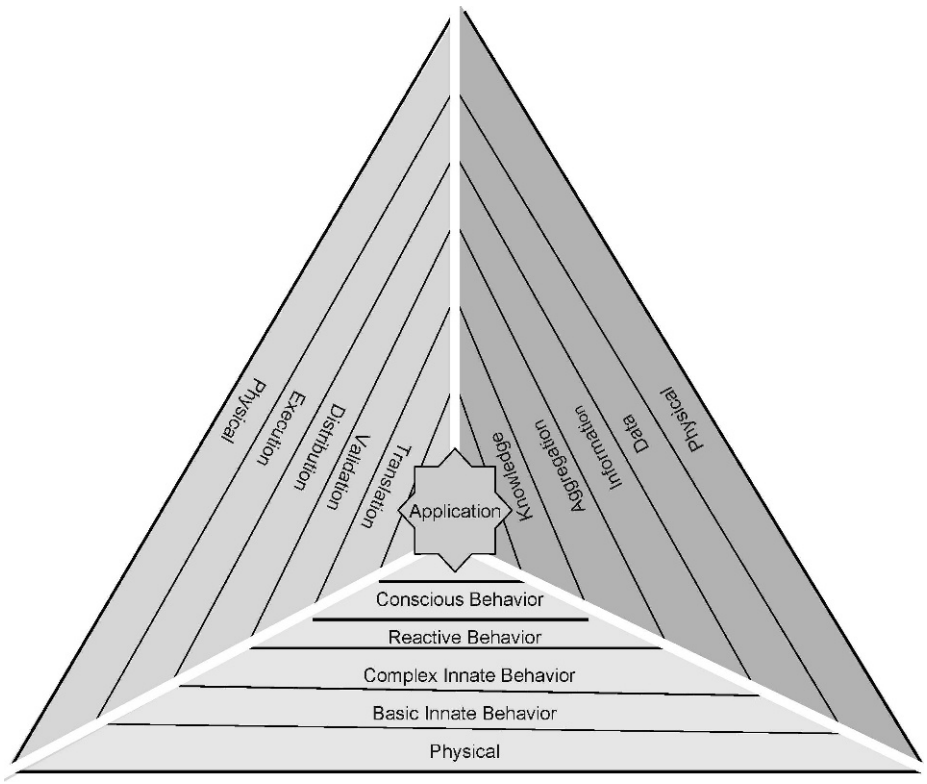

Figure 5. Integrated technical reference model

The conscious behavior layer. The conscious behavior layer provides schemas for checking goal validity and feasibility in the given situation by checking goals against intrinsic and extrinsic knowledge (global model) (Novales et al. 2006). It determines which goals should be accepted. It manages deliberative actions of the systems. In continuing our example, this layer would essentially perform a complete self-test when the system is powered up, and for example, report that all required hardware is operational and configured correctly, or possibly that the suggested synthetic instrument asked for could not be created because of limited bandwidth.

The application layer. The application layer is responsible for user interactions. It decides what information should be furnished to whom and when. It either consumes knowledge passed up to it (in the case of an intelligent program) or passes the knowledge to the user. It receives commands

The integrated technical reference model

The I-TRM illustrated in *Figure 5* is a refinement of architectural principals that have been suggested by Joshi and Michel (2007, 2008). It is an integrated view of the IC-face, C-face, and B-face.

The I-TRM has six layers: (a) physical; (b) data, execution, and basic innate behavior; (c) information, distribution, and complex innate behavior; (d) aggregation, translation, and reactive behavior; (e) knowledge, validation, and conscious behavior; and (f) application.

It is envisioned that each layer is a combination of hardware and/or software that can be specified through its interfaces, specifically its input and output data structures (data/metadata, information/meta-information, knowledge, status, control, and goal structures) and its ability to transform those data structures (behaviors).

Commercial devices that span one or more layers could be built with companies differentiating their products by the implementations behind the open specification of the interfaces. This is already happening with the adoption of the IEEE-1451 family of standards for smart sensors. These smart sensors can implement the lower two or three layers of the I-TRM. An equivalent effort to standardize the upper layers of the pyramid would allow companies to develop and market software products to simplify or automate test setup, operation, and data interfaces.

Sensor network architecture

Consider a sensor network architecture built using the principals of the I-TRM. Sensor networks are used for taking environmental measurements and are distributed systems. The proposed architecture is implemented as a three-tiered hierarchy. The bottom

tier consists of sensor nodes. The second tier is composed of cluster heads, and the third tier is composed of a single root node.

Sensor nodes are equipped with various sensors and are capable of performing basic networking, computing, and sensing tasks. A group of sensor nodes are connected through a local one-hop network to a cluster head, the next tier level.

Cluster heads are functionally more powerful units with more computational power, advanced data traffic and networking capabilities, and rich power resources for maintaining its one-hop communication with all subordinating sensor nodes and the root node. Cluster heads process data traffic and handle complex data processing to enrich the informational value associated with the data.

The root node is at the top of this hierarchy. The manner in which sensor nodes, cluster heads, and root node are connected is a tree arrangement. This arrangement provides an easy mechanism for efficient distribution of computational and power resources. This architectural arrangement is inspired by the architecture of a sensor network described in Stojmenovic (2005) and the IEEE 1451 standards. It was proposed in Joshi and Michel (2008) and by Joshi (2008) in his thesis. The reader is referred to these references for a description of the various class diagrams associated with this software architecture.

Sensor node description

A sensor node is the most repetitive physically instantiated unit. As discussed earlier, a sensor is the simplest of the functional units. Its functional blocks span the I-TRM layers 1 to 3.

Layer 1 is a physical layer that consists of sensors and mechanical units. This layer executes actions as directed by higher layers and possesses no intelligence of its own. It gathers raw data in unformatted, unverified, and transitory format. It deals with the electric, mechanical, and procedural characteristics of the system including the working of transducers. Layer 1 implements transducer working commands, transducer data, and basic sensor behaviors (sensor sensing, transmitting, sleeping, and hibernating). The transducer working commands are the set of commands for controlling the Layer 0 transducers, whereas transducer data are the output of the transducer. The basic sensor behaviors are the functionality or behaviors of the sensor that do not require any external information.

Layer 2 contains objects of more informational value and executes more complex commands and behaviors. Sensor data are created by calibrating transducer data. This processing is performed when the sensor is in the sensor-sensing state. Because this design is inspired by IEEE 1451 standards, the implementation of the transducer electronic data sheet (TEDS) is required. It allows for self-calibration of the transducer in the sensor. TEDS would be implemented in the system by introducing calibration data, a data structure. Self-calibration would be initiated with a transducer calibration command.

Cluster head description

A cluster head is a link between sensor nodes and the root node, and spans Layers 3 and 4. It is responsible for data aggregation and data consolidation. There are five different types of parallel running processes in this functional unit. Two types of processes are used to maintain correspondence between a sensor node and the cluster head, that is, one for issuing commands to a sensor node and a second for gathering information from a sensor node. The third process is responsible for performing data aggregation by combining data from various sensor nodes into a single data unit, cluster information. A fourth process controls the communication between the cluster head and root node. It is envisioned that these processes would be configurable through a user interface, the fifth process. A specific example of this would be a graphical or menu-driven user interface that allowed the test conductor to associate specific sensors to control aggregation to create synthetic instruments. Another example would be the ability of this aggregation process to respond to changes in the state of meta-information and adapt the lower-level system tasking to optimize system performance.

Within this architecture, there is a data object, the Sensor Instance Information object, related to each sensor. This object contains data from its associated sensor and the metadata related to that sensor. This metadata makes hot swapping possible. Processes, which maintain the correspondence between a sensor unit and a cluster head, are dedicated processes for each node, so each process has its own copy of every required component and runs independently of the other.

Root node description

The root node is at the top of the sensor network architecture hierarchy and spans I-TRM layers 5 and 6. It is a functional unit that performs knowledge extraction from aggregated information from various cluster heads, accepts and validates the test scenario, and runs the primary user interface software. It is situated at the base station.

The five main processes are:

1. maintaining correspondence between the root node and cluster heads
2. extracting knowledge from the aggregated information from the various cluster heads

3. updating the global knowledge base
4. validating tasking
5. providing a user interface

The processes for maintaining communication between the root nodes and various cluster head can be subcategorized into two groups, one for issuing commands to cluster heads and the second for gathering information from cluster heads. To have the validation feature, proxies for the root controller are generated for each application level process.

Conclusions

This article presents a framework for discussing, designing, and developing sensor network architectures based on modeling system components from three orthogonal abstractions: control distribution, information processing, and intelligence (behavior). This methodology can produce robust systems that are self-aware, self-healing and adaptable within a resource-constrained environment. It provides a potential mechanism for realizing interoperability between various subsystems—both hardware and software—across various manufacturers. ❑

HOWARD E. MICHEL retired from the U. S. Air Force in 1994, having served as a pilot, satellite launch director, engineer, and engineering manager. Dr. Michel earned his doctorate from Wright State University in 1999 and is currently Associate Professor of Electrical and Computer Engineering at the University of Massachusetts Dartmouth. He holds two patents and has published over 40 articles on intelligent systems, artificial neural networks, and optical computing. His research interests include autonomous vehicles, artificial neural networks, and distributed sensor networks. Dr. Michel is a senior member of the IEEE and IEEE Region 1 Director (2008–2009). E-mail: hmichel@umassd.edu

HEMANT JOSHI received his master of science in computer engineering from the University of Massachusetts Dartmouth. His master's thesis was entitled "Autonomous Mobile Sensor Networks Architecture for Hazard Detection and Surveillance." He received a bachelor's degree in engineering (Electronics & Communication Engineering) from Visvesvaraya Technological University, Belgaum, India. Mr. Joshi worked as a lecturer in the ECE Department, Maharishi Arvind Institute of Engineering, Jaipur, India and is currently employed by VMware as a development engineer. E-mail: g-hjoshi@umassd.edu

References

Albus, J. S., Barbera, A. J. and Nagel, R. N. 1981. "Theory and Practice of Hierarchical Control." *Proceedings of the 23rd IEEE Computer Society International Conference*. Washington, D. C. September 15–17.

Albus, J. S., Lumia, R., Fiala, J. and Wavering, A. 1989. "NASREM—The NASA/NBS Standard Reference Model for Telerobot Control System Architecture." *Proceedings of the 20th International Symposium on Industrial Robots*. Tokyo, Japan. October 4–6, 1989.

Albus, J. S. and Rippey, W. G. "RCS: A Reference Model Architecture for Intelligent Control." *Perception to Action Conference, 1994, Proceedings*. September 7–9, 1994, pp. 218–229.

Arkin, R. 1998. *Behavior-Based Robotics*. Cambridge, Massachusetts: MIT Press.

Barbera, A. J., Fitzgerald, M. L., and Albus, J. S. 1982. "Concepts for a Real-Time Sensory-Interactive Control System Architecture." *Proceedings of the Fourteenth Southeastern Symposium on System Theory*. Blacksburg, Virginia, April 15–16, 1982.

Day, J. D. and Zimmermann, H. 1983. "The OSI Reference Model," *Proceedings of the IEEE*. 71(12), pp. 1334–1340.

Dippel, H. A. and Michel, H. E. 2006. "The Control Technical Reference Model." *International Conference on Artificial Intelligence*, Las Vegas, Nevada, June 2006.

Fortier, P. and Michel, H. E. 2005. "Comparison of the EI TRM versus TENA," ITEA Technology Review Workshop, Atlanta, Georgia, July 12–14, 2005.

ICH Architecture Resource Centre. 2008. "Interoperability Clearing House Glossary of Terms" Available at http://www.ichnet.org/glossary.htm. Accessed August 28, 2008.

IEEE. 1999. *IEEE Std 1220-1998*. New York: Institute of Electrical and Electronics Engineers.

ISO. 1983. "Open system interconnection, basic research model, ISO International Standards IS 7498, ISO/TC 97/SCI6. Geneva, Switzerland: International Organization for Standards.

JAUS. 2008. Domain Model Volume I, Joint architecture for unmanned system. Available at http://www.jauswg.org/. Accessed August 28, 2008.

JAUS. 2008. Reference architecture specification volume II, part 1, architecture framework. *Joint Architecture for Unmanned System.* Available at http://www.jauswg.org/. Accessed August 28, 2008.

JAUS. 2008. Reference architecture specification volume II, part 2, message definition. *Joint Architecture for Unmanned Systems.* Available at http://www.jauswg.org. Accessed August 28, 2008.

JAUS. 2008. Reference architecture specification volume III, part 3, message set. *Joint Architecture for Unmanned Systems.* Available at http://www.jauswg.org. Accessed August 28, 2008.

Joshi, H. and Michel, H. E. 2007. "Integrating Information-Centric, Control-Centric and Behavior-Centric Technical Reference Models for Autonomous Sensor Networks." *International Conference on Wireless Networks.* Las Vegas, Nevada. June 2007.

Joshi, H. and Michel, H. E. 2008. "Integrated Technical Reference Model and Sensor Network Architecture." *International Conference on Wireless Networks*, Las Vegas, Nevada. June 2008.

Joshi, H. 2008. "Autonomous Mobile Sensor Networks Architecture for Hazard Detection and Surveillance," University of Massachusetts Dartmouth Master's thesis. January 2008.

Michel, H. E. and Fortier, P. 2005. "Development of an Embedded Instrumentation System Architecture and Its Comparison to the Test and Training Enabling Architecture." *Defense Transformation and Network-Centric Systems*, Proceedings of SPIE Vol. 6249. April 2005.

MOOS. 2008. *Mission-Oriented Operating Suite*. Available at http://www.robots.ox.ac.uk/~pnewman/TheMOOS/. Accessed August 28, 2008.

Nagrath, I. J. and Gopal, M. 1981. *Control System Engineering*, Second Edition. New Delhi: Wiley.

Novales, C., Mourioux, G. and Poisson, G. 2006. "A Multi-Level Architecture Controlling Robots from Autonomy to Teleoperation." *First National Workshop on Control Architectures of Robots*. Montpellier, Vermont. April 6, 2006.

Stojmenovic, I. 2005. *Sensor Networks Algorithms and Architecture*. New York: Wiley Publications.

Tanenbaum, A. S. 1996. *Computer Networks*, Third Edition. Upper Saddle River, New Jersey: Prentice Hall.

Visnevski, N. 2008. "Embedded Instrumentation Systems Architecture." *Instrumentation and Measurement Technology Conference Proceedings, IMTC 2008*. May 12–15, 2008, pp. 1134–1139.

Visnevski, N. and Bezdecny, M. 2008. "Test & Evaluation of Cognitive and Social Capabilities of Collaborative Unmanned Autonmous Systems." *ITEA Tech Review Conference*, Colorado Springs, Colorado. July 2008.

Acknowledgment

This work was partially supported by the Test and Resources Management Center's (TRMC) Test and Evaluation/Science and Technology (T&E/S&T) Program Non-Intrusive Instrumentation office through Naval Underwater Warfare Center (NUWC). The views and opinions expressed or implied in this article are those of the authors and not necessarily those of the Department of Defense or any of its subordinate agencies.